```
 1: // $Id: oclib.h,v 1.20 2018-11-08 14:39:15-08 - - $
 2:
 3: #ifndef __OCLIB_H__
 4: #define __OCLIB_H__
 5:
 6: #ifdef __OCLIB_C__
 7:
 8: typedef char* cstring;
 9: void* xcalloc (int nelem, int size);
10: void putint (int);
11: void putstr (const cstring);
12: cstring getword (void);
13: cstring getln (void);
14: void endl();
15:
16: #else
17:
18: #define EXIT_SUCCESS 0
19: #define EXIT_FAILURE 1
20: #define EOF (-1)
21: #define char int
22: #define bool int
23: #define true 1
24: #define false 0
25: #define assert(expr) { \
26:            if (not (expr)) { \
27:               __assert_fail (#expr, __FILE__, __LINE__, __FUNC__); \
28:            } \
29:         }
30: void __assert_fail (string expr, string file, int line, string func);
31: void putchar (char c);
32: void putint (int i);
33: void putstr (string s);
34: int getchar();
35: string getword();
36: string getln();
37: void endl();
38: void exit (int status);
39:
40: #endif
41:
42: #endif
43:
```

```
 1: # 1 "oclib.h"
 2: # 1 "<built-in>"
 3: # 1 "<command-line>"
 4: # 1 "/usr/include/stdc-predef.h" 1 3 4
 5: # 1 "<command-line>" 2
 6: # 1 "oclib.h"
 7:
 8:
 9:
10:
11:
12:
13:
14: typedef char* cstring;
15: void* xcalloc (int nelem, int size);
16: void putint (int);
17: void putstr (const cstring);
18: cstring getword (void);
19: cstring getln (void);
20: void endl();
```

```
# 1 "oclib.h"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "oclib.h"
# 30 "oclib.h"
void __assert_fail (string expr, string file, int line, string func);
void putchar (int c);
void putint (int i);
void putstr (string s);
int getchar();
string getword();
string getln();
void endl();
void exit (int status);
```

```
 1: // $Id: oclib.c,v 1.88 2018-11-08 15:15:06-08 - - $
 2:
 3: #include <assert.h>
 4: #include <ctype.h>
 5: #include <stdio.h>
 6: #include <stdlib.h>
 7: #include <string.h>
 8:
 9: #define __OCLIB_C__
10: #include "oclib.h"
11:
12: void* xcalloc (int nelem, int size) {
13:    void* result = calloc (nelem, size);
14:    assert (result != NULL);
15:    return result;
16: }
17:
18: typedef int (*ctype_fn) (int);
19: cstring scan (ctype_fn skip_over, ctype_fn stop_at) {
20:    int byte = 0;
21:    do {
22:       byte = getchar();
23:       if (byte == EOF) return NULL;
24:    } while (skip_over != NULL && skip_over (byte));
25:    size_t buf_size = 16;
26:    cstring buffer = malloc (buf_size);
27:    assert (buffer != NULL);
28:    size_t end_pos = 0;
29:    do {
30:       buffer[end_pos++] = byte;
31:       if (end_pos >= buf_size) {
32:          buf_size *= 2;
33:          buffer = realloc (buffer, buf_size);
34:          assert (buffer != NULL);
35:       }
36:       buffer[end_pos] = '\0';
37:       byte = getchar();
38:    }while (byte != EOF && !stop_at (byte));
39:    return buffer;
40: }
41:
42: int isnl (int byte)           { return byte == '\n'; }
43: void putint (int val)         { printf ("%d", val); }
44: void putstr (const cstring s) { printf ("%s", s); }
45: cstring getword (void)        { return scan (isspace, isspace); }
46: cstring getln (void)          { return scan (NULL, isnl); }
47: void endl()                   { putchar ('\n'); }
48:
```

```
 1: 0000000000000040 r __PRETTY_FUNCTION__.3060
 2: 0000000000000048 r __PRETTY_FUNCTION__.3072
 3:                  U __assert_fail
 4:                  U calloc
 5: 00000000000001e9 T endl
 6:                  U getchar
 7: 00000000000001d4 T getln
 8: 00000000000001bf T getword
 9: 0000000000000165 T isnl
10:                  U isspace
11:                  U malloc
12:                  U printf
13:                  U putchar
14: 0000000000000178 T putint
15: 000000000000019a T putstr
16:                  U realloc
17: 000000000000004e T scan
18: 0000000000000000 T xcalloc
```